| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/755,389 | 01/05/2001 | Sanjeev Banerjia | 10990960-1 | 5215 |

22879    7590    12/16/2005

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

| EXAMINER |
|---|
| FOWLKES, ANDRE R |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

DATE MAILED: 12/16/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

| | **Application No.** | **Applicant(s)** |
| --- | --- | --- |
| | 09/755,389 | BANERJIA ET AL. |
| ***Office Action Summary*** | **Examiner** | **Art Unit** | |
| | Andre R. Fowlkes | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *23 September 2005*.

2a)☒ This action is **FINAL**.  2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-23* is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-23* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All  b)☐ Some * c)☐ None of:

        1.☐ Certified copies of the priority documents have been received.

        2.☐ Certified copies of the priority documents have been received in Application No. _____.

        3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1)☒ Notice of References Cited (PTO-892)

2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3)☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
    Paper No(s)/Mail Date _____.

4)☐ Interview Summary (PTO-413)
    Paper No(s)/Mail Date. _____.

5)☐ Notice of Informal Patent Application (PTO-152)

6)☐ Other: _____.

## DETAILED ACTION

1.     This action is in response to the amendment filed 9/23/05.

2.     Claims 1-6, 10, 12-13, 15-18, 20 and 22-23 have been amended and no claims

have been canceled or newly added.

### *Claim Rejections - 35 USC § 103*

3.     The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set
> forth in section 102 of this title, if the differences between the subject matter sought to be patented and
> the prior art are such that the subject matter as a whole would have been obvious at the time the
> invention was made to a person having ordinary skill in the art to which said subject matter pertains.
> Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-10, 12-20, 22 and 23 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Buzbee, U.S. Patent No. 5,815,720 (art made of record), in view of

Chilimbi et al. (Chilimbi), U.S. Patent No. 6,330,556.

As per claim 1, Buzbee discloses **a method for operating a code cache in a**

**dynamic instruction translator,** (col. 3:5-29, "FIG. 1 shows a simplified block diagram

of a computing system 7 which uses a dynamic translator 15 to execute an application

10. Dynamic translator 15 differs from a compiler in that a compiler produces executable

code before runtime. Dynamic translator 15 produces translated code 8 during runtime.

Blocks of code from application 10 are translated at execution time. The translated

blocks of code are shown in FIG. 1 as translated code 8. Translated code 8 is stored in

memory (e.g., cache memory) so that each block of code which is executed a number

of times need be translated only once during runtime. This approach allows the flexibility

of not requiring that code be translated before runtime, but reduces the overhead that

would result if a block of code were translated every time it was executed.

(3)   In the preferred embodiment, translated code 8 is stored in a translated code cache

20. When translated code cache 20 is full, it may be necessary to discard some

previously translated blocks of code in order to make room for newly translated blocks

of code. This will require the blocks of code which were discarded to be retranslated if

they are used again, but does allow for potential savings in memory usage. Alternately,

rather than discarding previously translated blocks of code, they can be stored in

system memory"),   **comprising:**

**- storing a plurality of instruction translations in a ... code cache memory**

**that only stores instructions translations generated by a dynamic instruction**

**translator** (col. 3:5-29, "FIG. 1 shows a simplified block diagram of a computing system

7 which uses a dynamic translator 15 to execute an application 10. Dynamic translator

15 differs from a compiler in that a compiler produces executable code before runtime.

Dynamic translator 15 produces translated code 8 during runtime. Blocks of code from

application 10 are translated at execution time. The translated blocks of code are shown

in FIG. 1 as translated code 8. Translated code 8 is stored in memory (e.g., cache

memory) so that each block of code which is executed a number of times need be

translated only once during runtime. This approach allows the flexibility of not requiring

that code be translated before runtime, but reduces the overhead that would result if a

block of code were translated every time it was executed.

In the preferred embodiment, <u>translated code 8 is stored in a translated code</u>

<u>cache 20</u>. When translated code cache 20 is full, it may be necessary to discard some

previously translated blocks of code in order to make room for newly translated blocks

of code. This will require the blocks of code which were discarded to be retranslated if

they are used again, but does allow for potential savings in memory usage. Alternately,

rather than discarding previously translated blocks of code, they can be stored in

system memory").


Buzbee does not explicitly disclose:

- storing a plurality of instruction translations **in a cold partition** in a code cache

memory,

**- determining whether the instruction translation that has been stored in**

**the cold partition is hot,**

**- moving the instruction translation to a hot partition in the code cache**

**memory when an instruction translation has been determined to be hot**.


However, Chilimbi, in an analogous environment, discloses:

- storing a plurality of instruction translations **in a cold partition** in a code cache

memory (col. 2 lines 39-43, "the most heavily referenced (data) ... are kept in a hot

(memory location) ... while the remaining (data) ... are placed in a ... cold (memory

location)"),

    **- determining whether the** instruction translation **that has been stored in the**

**cold partition is hot** (col. 2 lines 39-43, "the most heavily referenced (data) ... are kept

in a hot (memory location) ... while the remaining (data) ... are placed in a ... cold

(memory location)"),

    **- moving the** instruction translation **to a hot partition in the code cache**

**memory when an** instruction translation **has been determined to be hot** (col. 2 lines

37-43, "The partitioning is based on profile information about (data) ... access counts ...

the most heavily referenced (data are placed) ... in a hot (memory location) ... while the

remaining (data) ... are placed in a ... cold (memory location)").


    Therefore, it would have been obvious to a person of ordinary skill in the art, at

the time the invention was made, to incorporate the teachings of Chilimbi into the

system of Buzbee to have:

    **- storing a plurality of instruction translations in a cold partition in a code**

**cache memory,**

    **- determining whether the instruction translation that has been stored in**

**the cold partition is hot,**

    **- moving the instruction translation to a hot partition in the code cache**

**memory when an instruction translation has been determined to be hot.** The

modification would have been obvious because one of ordinary skill in the art would

have wanted to fully exploit the performance advantages of using the well-known and

well documented hot/cold cache for instruction translations and any other type of

code/data that would benefit from an optimized caching scheme, (Chilimbi, 1:49-2:56

and 18:64-19:59).


As per claim 2, the rejection of claim 1 is incorporated and further, Buzbee

doesn't explicitly disclose that **the step of determining whether the instruction**

**translation is hot comprises: maintaining a different associated counter for each**

**of a plurality of instruction translation in the cold partition of the code cache**

**memory; incrementing or decrementing the count in the associated counter each**

**time its associated instruction translations is executed; and concluding the**

**determination that a instruction translations is hot if the count in the associated**

**counter reaches a first threshold value.**

However, Chilimbi, in an analogous environment, discloses that **the step of**

**determining whether the instruction translation is hot comprises: maintaining a**

**different associated counter for each of a plurality of instruction translation in the**

**cold partition of the code cache memory; incrementing or decrementing the**

**count in the associated counter each time its associated instruction translations**

**is executed; and concluding the determination that a instruction translations is**

**hot if the count in the associated counter reaches a first threshold value** (col. 2

lines 37-43, "The partitioning is based on profile information about (instruction

translations) ... access counts ... the most heavily (executed instruction translations) ...

are kept in a hot (memory location) ... while the remaining (instruction translations) ...

are placed in a ... cold (memory location)").

Therefore, it would have been obvious to a person of ordinary skill in the art, at

the time the invention was made, to incorporate the teachings of Chilimbi into the

system of Buzbee to have **the step of determining whether the instruction**

**translation is hot comprises: maintaining a different associated counter for each**

**of a plurality of instruction translation in the cold partition of the code cache**

**memory; incrementing or decrementing the count in the associated counter each**

**time its associated instruction translations is executed; and concluding the**

**determination that a instruction translations is hot if the count in the associated**

**counter reaches a first threshold value.** The modification would have been obvious

because one of ordinary skill in the art would have wanted to fully exploit the

performance advantages of using the well-known and well documented hot/cold cache

for instruction translations and any other type of code/data that would benefit from an

optimized caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).


As per claim 3, the rejection of claim 1 is incorporated and further, Buzbee

doesn't explicitly disclose that the **hot partition is contiguous and disjoint from said**

**cold partition in said code cache memory.**

However, Chilimbi, in an analogous environment, discloses that the **hot**

**partition is contiguous and disjoint from said cold partition in said code cache**

**memory** (col. 3 lines 65-67, "division of (instruction translations) into two (groups)

comprising hot access fields and cold access fields", and Fig. 1, item 22, and the

associated text (e.g. col. 4 line 5 – col. 5 line 67), describes the computer memory used

in this system, which is capable of containing contiguous and disjoint memory sections).

Therefore, it would have been obvious to a person of ordinary skill in the art, at

the time the invention was made, to incorporate the teachings of Chilimbi into the

system of Buzbee to have a **hot partition that is contiguous and disjoint from said**

**cold partition in said code cache memory**. The modification would have been

obvious because one of ordinary skill in the art would have wanted to fully exploit the

performance advantages of using the well-known and well documented hot/cold cache

for instruction translations and any other type of code/data that would benefit from an

optimized caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).


As per claim 4, the rejection of claim 2, is incorporated and further, Buzbee

doesn't explicitly disclose **maintaining an associated counter step comprises**

**maintaining counters in a data structure external to code cache memory**.

However, Chilimbi, in an analogous environment, discloses **maintaining an**

**associated counter step comprises maintaining counters in a data structure**

**external to code cache memory** (col. 2 lines 37-39, "The partitioning is based on

profile information about (instruction translations) ... access counts", and Fig. 1, item 20,

and the associated text (e.g. col. 4 line 5 – col. 5 line 67), describes a conventional

computer system, which is capable of maintaining counters external to cache memory).

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Chilimbi into the system of Buzbee to have **maintaining an associated counter step comprises maintaining counters in a data structure external to code cache memory**. The modification would have been obvious because one of ordinary skill in the art would have wanted to fully exploit the performance advantages of using the well-known and well documented hot/cold cache for instruction translations and any other type of code/data that would benefit from an optimized caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).

As per claim 5, the rejection of claim 4 is incorporated and further, Buzbee doesn't explicitly disclose **the step of at least temporarily delinking blocks of translations stored in said cold partition so that control exits the code cache memory in order to perform the incrementing or decrementing step**.

However, Chilimbi, in an analogous environment, discloses **the step of at least temporarily delinking blocks of translations stored in said cold partition so that control exits the code cache memory in order to perform the incrementing or decrementing step** (col. 2 lines 35-36, "Data structures (blocks) are partitioned (delinked and accounted for) into heavily referenced and less heavily references partitions").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Chilimbi into the

system of Buzbee to have **the step of at least temporarily delinking blocks of translations stored in said cold partition so that control exits the code cache memory in order to perform the incrementing or decrementing step**. The modification would have been obvious because one of ordinary skill in the art would have wanted to fully exploit the performance advantages of using the well-known and well documented hot/cold cache for instruction translations and any other type of code/data that would benefit from an optimized caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).

As per claim 6, the rejection of claim 2 is incorporated and further, Buzbee doesn't explicitly disclose that **maintaining within said memory an associated counter step comprises maintaining one of said associated counters for each entry point into a plurality of the instruction translations in said cold partition of the code cache memory**.

However, Chilimbi, in an analogous environment, discloses that **maintaining within said memory an associated counter step comprises maintaining one of said associated counters for each entry point into a plurality of the instruction translations in said cold partition of the code cache memory** (col. 2 lines 37-39, "The partitioning is based on profile information (maintained in the memory) about (instruction translations) … access counts").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Chilimbi into the

system of Buzbee to have **maintaining within said memory an associated counter step comprises maintaining one of said associated counters for each entry point into a plurality of the instruction translations in said cold partition of the code cache memory.** The modification would have been obvious because one of ordinary skill in the art would have wanted to fully exploit the performance advantages of using the well-known and well documented hot/cold cache for instruction translations and any other type of code/data that would benefit from an optimized caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).

As per claim 7, the rejection of claim 2 is incorporated and further, Buzbee doesn't explicitly disclose **maintaining an associated counter step comprises logically embedding update code on an arc between two instruction translations.**

However, Chilimbi, in an analogous environment, discloses **maintaining an associated counter step comprises logically embedding update code on an arc between two instruction translations,** items (col. 6 lines 58-63, "Each of the data elements defined in FIG. 2 are shown as nodes (i.e. individual data items) in FIG. 3 with arcs or edges drawn between them. The nodes are representative of all instances of the data structure. The edges are weighted to indicate field affinity (i.e. an associated counter)").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Chilimbi into the system of Buzbee to have **maintaining an associated counter step comprises**

**logically embedding update code on an arc between two instruction translations.**
The modification would have been obvious because one of ordinary skill in the art would have wanted to fully exploit the performance advantages of using the well-known and well documented hot/cold cache for instruction translations and any other type of code/data that would benefit from an optimized caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).

As per claim 8, the rejection of claim 2 is incorporated and further, Buzbee doesn't explicitly disclose **maintaining an associated counter step comprises maintaining one of said associated counters for each** item in the hot and cold memory locations **in an associated microprocessor.**

However, Chilimbi, in an analogous environment, discloses **maintaining an associated counter step comprises maintaining one of said associated counters for each** item in the hot and cold memory locations **in an associated microprocessor** (col. 2 lines 37-43, "The partitioning is based on profile information about (instruction translations) ... access counts ... the most heavily referenced (instruction translations) ... are kept in a hot (memory location) ... while the remaining (instruction translations) ... are placed in a ... cold (memory location)").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Chilimbi into the system of Buzbee to have **maintaining an associated counter step comprises maintaining one of said associated counters for each** item in the hot and cold

memory locations **in an associated microprocessor**. The modification would have

been obvious because one of ordinary skill in the art would have wanted to fully exploit

the performance advantages of using the well-known and well documented hot/cold

cache for instruction translations and any other type of code/data that would benefit

from an optimized caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).


As per claim 9, the rejection of claim 2 is incorporated and further, Buzbee

doesn't explicitly disclose that the instruction translations **moving step comprises**

**sampling a plurality of said associated counters on an intermittent basis to**

**determine if the count therein has reached said threshold value**.

However, Chilimbi, in an analogous environment, discloses that the instruction

translations **moving step comprises sampling a plurality of said associated**

**counters on an intermittent basis to determine if the count therein has reached**

**said threshold value** (col. 2 lines 37-43, "The partitioning is based on profile

information about (instruction translations) ... access counts (which is sampled on an

intermittent basis, then) ... the most heavily (executed instruction translations) ... are

(placed) in a hot (memory location) ... while the remaining (instruction translations) ...

are placed in a ... cold (memory location)").

Therefore, it would have been obvious to a person of ordinary skill in the art, at

the time the invention was made, to incorporate the teachings of Chilimbi into the

system of Buzbee to have **moving step comprises sampling a plurality of said**

**associated counters on an intermittent basis to determine if the count therein has**

**reached said threshold value.** The modification would have been obvious because

one of ordinary skill in the art would have wanted to fully exploit the performance

advantages of using the well-known and well documented hot/cold cache for instruction

translations and any other type of code/data that would benefit from an optimized

caching scheme, (Chilimbi, 1:49-2:56 and 18:64-19:59).


As per claim 10, the rejection of claim 1 is incorporated and further, (Buzbee

doesn't explicitly disclose **determining if a number of hot instruction translations in**

**said hot partition of said code cache memory exceeds a second threshold value;**

**and if said number of said hot instruction translations exceeds said second**

**threshold value, then expanding the size of said hot partition in said code cache**

**memory by adding thereto an expansion area contiguous to said hot partition.**

However, Chilimbi, in an analogous environment,)Buzbee discloses **determining**

**if a number of hot instruction translations in said hot partition of said code cache**

**memory exceeds a second threshold value; and if said number of said hot**

**instruction translations exceeds said second threshold value, then expanding the**

**size of said hot partition in said code cache memory by adding thereto an**

**expansion area contiguous to said hot partition** (col. 2 lines 37-39, "The partitioning

is based on profile information about (instruction translations) ... access counts", and

Fig. 1, item 20, and the associated text (e.g. col. 4 line 5 – col. 5 line 67), describes a

conventional computer system, which creates and modifies numerous memory

partitions).

Therefore, it would have been obvious to a person of ordinary skill in the art, at

the time the invention was made, to incorporate the teachings of Chilimbi into the

system of Buzbee to have a **determining if a number of hot instruction translations**

**in said hot partition of said code cache memory exceeds a second threshold**

**value; and if said number of said hot instruction translations exceeds said**

**second threshold value, then expanding the size of said hot partition in said code**

**cache memory by adding thereto an expansion area contiguous to said hot**

**partition.** The modification would have been obvious because one of ordinary skill in

the art would have wanted to fully exploit the performance advantages of using the well-

known and well documented hot/cold cache for instruction translations and any other

type of code/data that would benefit from an optimized caching scheme, (Chilimbi, 1:49-

2:56 and 18:64-19:59).

As per claim 12, the Buzbee/Chilimbi also discloses such claimed limitations as

addressed in claim 2, above.

As per claims 13-20 and 22 this is a system version of the claimed method

discussed above, in claims 1-6, 9, 10 and 12, wherein all claimed limitations have also

been addressed and/or cited as set forth above. For example, see the Buzbee/Chilimbi

combination, (Buzbee, col. 3:8-25 and Chilimbi, col. 2:35-56).

As per claim 23, this is a product version of the claimed method discussed

above, in claim 1, wherein all claimed limitations have also been addressed and/or cited

as set forth above.  For example, see the Buzbee/Chilimbi combination, (Buzbee, col.

3:8-25 and Chilimbi, col. 2:35-56).


4.      Claims 11 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable

over Buzbee, U.S. Patent No. 5,815,720 (art made of record), in view of Chilimbi et al.

(Chilimbi), U.S. Patent No. 6,330,556, further in view of Walls, U.S. Patent No.

5,675,790.


As per claim 11, the rejection of claim 10 is incorporated and further the

Buzbee/Chilimbi combination does not explicitly disclose **removing all cold**

**translations from said expansion area and storing said removed translations in**

**said cold partition.**

However, Walls, in an analogous environment, discloses **removing all** less-

desirable data entries from a dynamic memory **area and storing said removed**

instruction translations in a separate location (col. 8 lines 36-39, "If the segment

(instruction translations) is smaller than the minimum size (less-desirable) then remove

the segment from the (section of) dynamic memory ... (and) insert the segment into a

separate (location)").

Therefore, it would have been obvious to a person of ordinary skill in the art, at

the time the invention was made, to incorporate the teachings of Walls into the

Buzbee/Chilimbi system to enable **removing all cold translations from said**

**expansion area and storing said removed translations in said cold partition.** The

modification would have been obvious because one of ordinary skill in the art would

want to maintain the temporal access advantages by keeping the less-desirable data

items together and separate from both the most and least desirable data items.

As per claim 21, the Buzbee/Chilimbi/Walls combination also discloses such

claimed limitations as addressed in claim 11, above.

### Response to Arguments

5.       Applicant's arguments with respect to claims 1, 10, 13, 20 & 23 have been

considered but are moot in view of the new ground(s) of rejection.

### Conclusion

6.       Applicant's amendment necessitated the new ground(s) of rejection presented in

this Office action.  Accordingly, **THIS ACTION IS MADE FINAL.**  See MPEP

§ 706.07(a).  Applicant is reminded of the extension of time policy as set forth in 37

CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE

MONTHS from the mailing date of this action.  In the event a first reply is filed within

TWO MONTHS of the mailing date of this final action and the advisory action is not

mailed until after the end of the THREE-MONTH shortened statutory period, then the

shortened statutory period will expire on the date the advisory action is mailed, and any

extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later

than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the

examiner should be directed to Andre R. Fowlkes whose telephone number is (571)

272-3697. The examiner can normally be reached on Monday - Friday, 8:00am-

4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for

the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the

Patent Application Information Retrieval (PAIR) system. Status information for

published applications may be obtained from either Private PAIR or Public PAIR.

Status information for unpublished applications is available through Private PAIR only.

For more information about the PAIR system, see http://pair-direct.uspto.gov. Should

you have questions on access to the Private PAIR system, contact the Electronic

Business Center (EBC) at 866-217-9197 (toll-free).

TUAN DAM
SUPERVISORY PATENT EXAMINER

ARF